Frontdoor Programming Assignment

Nick Miner

8/1/2019

Reading in data, importing packages and setting seed. Creating new features for diamond volume and the log price of the diamond. The 4 most important characteristics for determining the cost of a diamond are caret, cut, color and clarity. The data does not include the number of carets per diamond. To work around this, we use the x, y, and z features to create the volume of each diamond as feature. The volume of a diamond directly affects its weight, and the weight of a diamond plays a large factor in its price. In this way a missing feature can be substituted.
The log price is a a normalization feature since the price of a diamond can vary greatly. Predictions will be made with both prices in order to establish a better idea of correlation.

```
knitr::opts_chunk$set(echo = TRUE)
set.seed(0351)
library(caret)
```

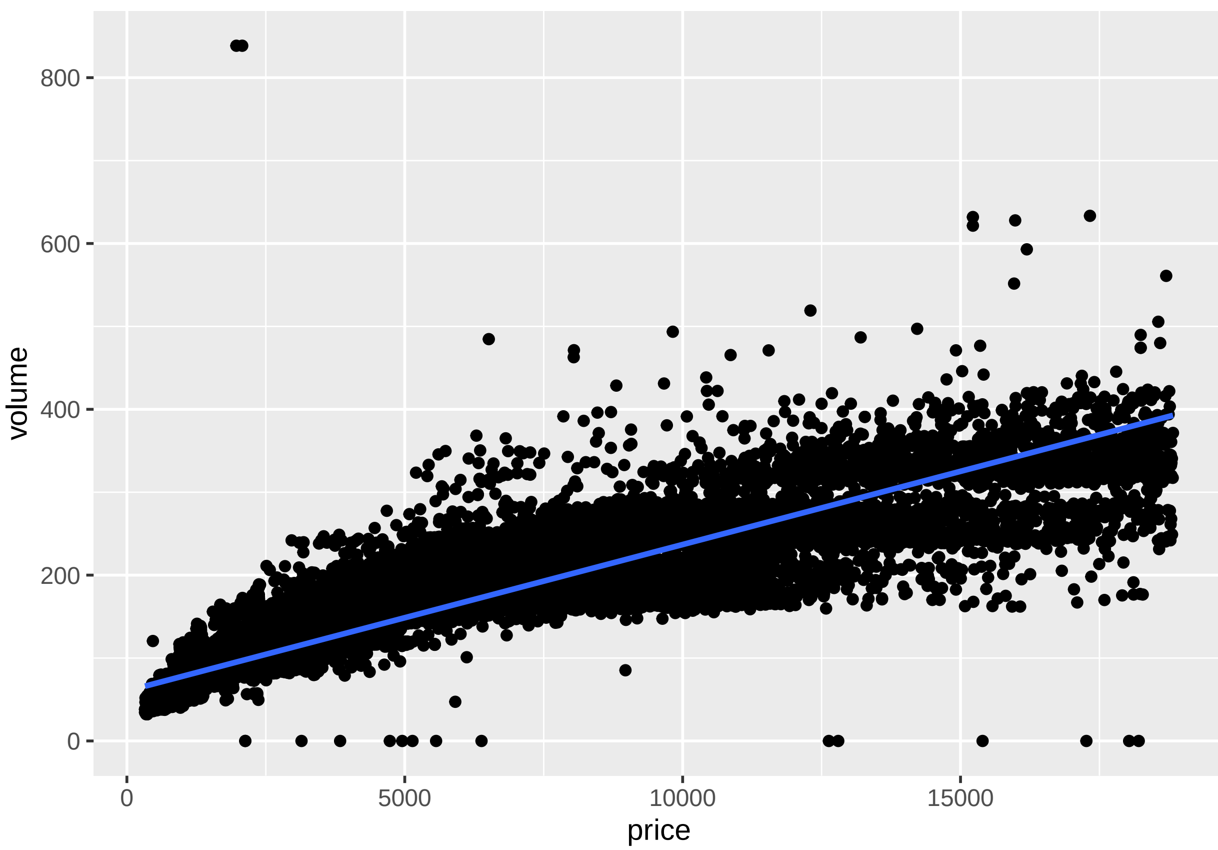
```
## Warning: package 'caret' was built under R version 3.6.1

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.6.1
```

```
library(ggplot2)
diamonds <- read.csv("C:/Users/nickm/MLinR/DiamondsPredictions/Diamonds.csv")
diamonds <- as.data.frame(diamonds)
diamonds$volume <- diamonds$x * diamonds$y * diamonds$z
diamonds$`log price` <- log10(diamonds$price)
```

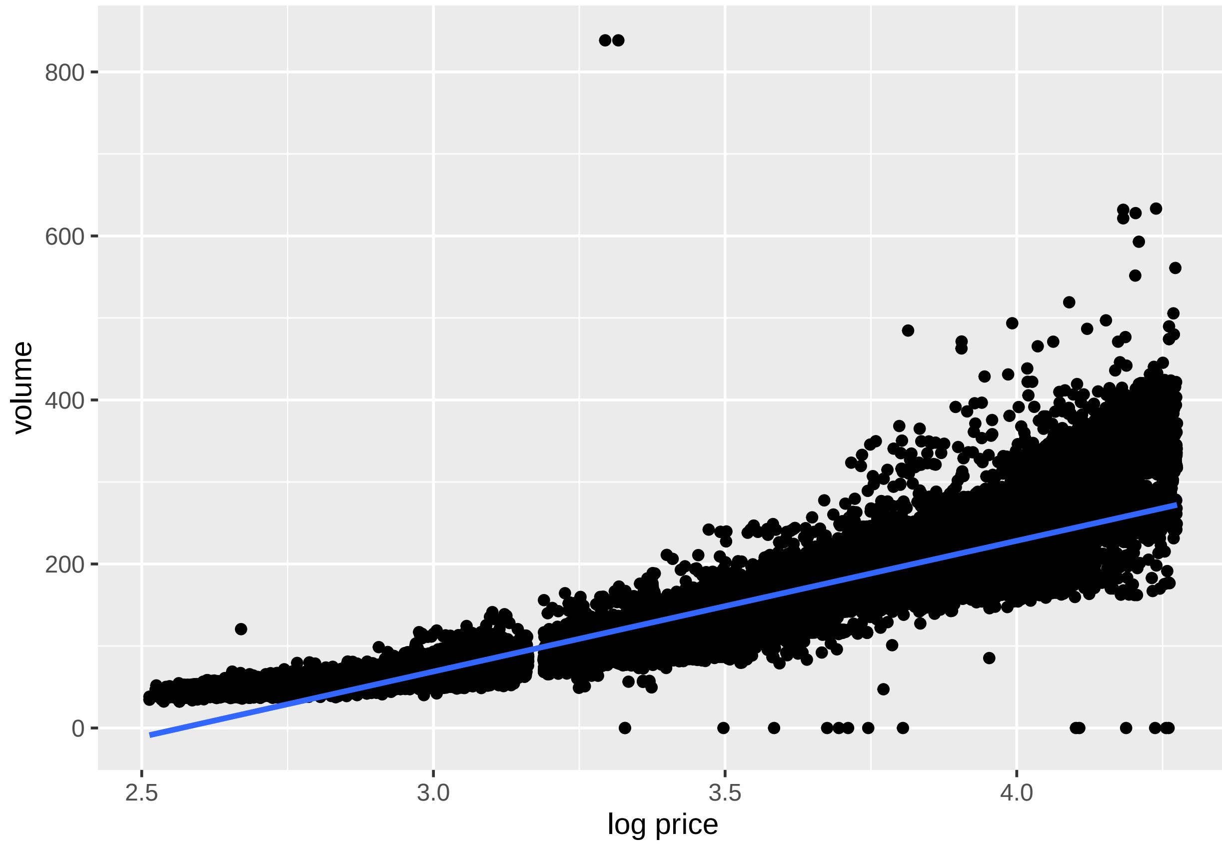
Exploratory data visualization. Plotting the color, cut and clarity against the price did not yield any linearity, suggesting the diamond pricing within categories is relatively fixed. Plotting the price against the volume of the diamond, however, suggests a more correlational relationship than any of the other features.

Plotting the log price against the diamond's volume reinforces this idea, as normalizing the price makes clear an even more linear relationship.

```
ggplot(diamonds, aes(price, volume)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)
```




```
ggplot(diamonds, aes(`log price`, volume)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)
```



Splitting the data into 80% training and 20% testing data. Removing features that do not explain a significant amount of variance in the data or that were used to calculate volume.

```
data_sample <- floor(0.8 * nrow(diamonds))
index <- sample(seq_len(nrow(diamonds)), size = data_sample)
diamonds_train <- diamonds[index,]
diamonds_test <- diamonds[-index,]

diamonds_train <- diamonds_train[, c(-5, -6, -7, -8, -9)]
diamonds_test <- diamonds_test[, c(-5, -6, -7, -8, -9)]
```

Setting a linear model to measure the effects of the features on the price of the diamond. The summary for the log price suggests that cut does not play a large role in the diamond's price; the difference in variance confirms this. The log model is then updated to remove the cut feature.

```
diamonds_lm <- lm(price ~ cut + color + clarity + volume,
                  data = diamonds_train)
summary(diamonds_lm)
```

```
##
## Call:
## lm(formula = price ~ cut + color + clarity + volume, data = diamonds_train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -41458   -681   -199    440  22758
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -6906.1544    71.2940  -96.87   <2e-16 ***
## cutGood        472.9736    46.2244   10.23   <2e-16 ***
## cutIdeal       662.1874    42.0521   15.75   <2e-16 ***
## cutPremium     612.7962    42.4024   14.45   <2e-16 ***
## cutVery Good   593.8625    42.9180   13.84   <2e-16 ***
## colorE        -244.4200    25.4608   -9.60   <2e-16 ***
## colorF        -296.9286    25.7149  -11.55   <2e-16 ***
## colorG        -472.8490    25.1856  -18.77   <2e-16 ***
## colorH        -908.2424    26.7270  -33.98   <2e-16 ***
## colorI       -1410.8053    30.0786  -46.90   <2e-16 ***
## colorJ       -2258.8599    36.8883  -61.23   <2e-16 ***
## clarityIF     5214.4944    72.4008   72.02   <2e-16 ***
## claritySI1    3409.8377    62.1965   54.82   <2e-16 ***
## claritySI2    2477.2105    62.4531   39.66   <2e-16 ***
## clarityVS1    4342.2463    63.4504   68.44   <2e-16 ***
## clarityVS2    4053.8432    62.4911   64.87   <2e-16 ***
## clarityVVS1   4847.1150    67.1138   72.22   <2e-16 ***
## clarityVVS2   4745.0318    65.3042   72.66   <2e-16 ***
## volume          54.4333     0.1028  529.25   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1231 on 31981 degrees of freedom
## Multiple R-squared:  0.9042, Adjusted R-squared:  0.9041
## F-statistic: 1.677e+04 on 18 and 31981 DF,  p-value: < 2.2e-16
```

```
diamonds_log_lm <- lm(`log price` ~ cut + color + clarity + volume,
                      data = diamonds_train)
summary(diamonds_log_lm)
```

```
##
## Call:
## lm(formula = `log price` ~ cut + color + clarity + volume, data = diamonds_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.3354 -0.0946  0.0243  0.1066  1.7765
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.374e+00  8.770e-03 270.745  < 2e-16 ***
## cutGood       4.270e-03  5.686e-03   0.751    0.453
## cutIdeal      4.694e-03  5.173e-03   0.907    0.364
## cutPremium    9.842e-04  5.216e-03   0.189    0.850
## cutVery Good -2.146e-05  5.279e-03  -0.004    0.997
## colorE       -2.492e-02  3.132e-03  -7.958 1.81e-15 ***
## colorF       -1.971e-02  3.163e-03  -6.232 4.66e-10 ***
## colorG       -5.603e-02  3.098e-03 -18.086  < 2e-16 ***
## colorH       -1.096e-01  3.288e-03 -33.328  < 2e-16 ***
## colorI       -1.762e-01  3.700e-03 -47.620  < 2e-16 ***
## colorJ       -2.454e-01  4.538e-03 -54.077  < 2e-16 ***
## clarityIF     4.161e-01  8.906e-03  46.723  < 2e-16 ***
## claritySI1    2.889e-01  7.651e-03  37.756  < 2e-16 ***
## claritySI2    2.138e-01  7.682e-03  27.826  < 2e-16 ***
## clarityVS1    3.523e-01  7.805e-03  45.131  < 2e-16 ***
## clarityVS2    3.321e-01  7.687e-03  43.207  < 2e-16 ***
## clarityVVS1   3.797e-01  8.256e-03  45.989  < 2e-16 ***
## clarityVVS2   3.744e-01  8.033e-03  46.605  < 2e-16 ***
## volume        5.877e-03  1.265e-05 464.546  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1514 on 31981 degrees of freedom
## Multiple R-squared:  0.8819, Adjusted R-squared:  0.8818
## F-statistic: 1.326e+04 on 18 and 31981 DF,  p-value: < 2.2e-16
```

```r
diamonds_log_lm <- lm(`log price` ~ color + clarity + volume,
                      data = diamonds_train)
```

Predictions for the testing set are made here. The linear model model for price trained on the training data is set to the test data and used to predict price points for the testing diamonds. The absolute value of those points (the diamond price can't be negative) is then fed into a correlation matrix. The same is done for predictions based on the log price for the diamonds.

The correlation matrix for each linear model presents a clear view that the predicted values for both the price and log price are very accurate! Most of the diamond price points correlate very strongly with the the fit, upper and lower bounds - all the correlation values are at least 0.95.

Interestingly, the log price linear model showed even more correlation than the price model, despite having one less feature. The logging of the prices must have reduced variance for more accurate predictions.

```r
lm_predictions <- predict(diamonds_lm, newdata = diamonds_test,
                          interval = "prediction", level = 0.95)
test_preds <- data.frame(cbind(actuals=diamonds_test$price,
                                  predicteds=lm_predictions))
test_preds$fit <- abs(test_preds$fit)

correlation_accuracy <- cor(test_preds)
correlation_accuracy
```

```
##           actuals       fit       lwr       upr
## actuals 1.0000000 0.9638806 0.9580998 0.9581044
## fit     0.9638806 1.0000000 0.9910385 0.9910459
## lwr     0.9580998 0.9910385 1.0000000 1.0000000
## upr     0.9581044 0.9910459 1.0000000 1.0000000
```

```r
lm_log_predictions <- predict(diamonds_log_lm, newdata = diamonds_test,
                          interval = "prediction", level = 0.95)
test_log_preds <- data.frame(cbind(actuals=diamonds_test$price,
                                  predicteds=lm_log_predictions))
test_log_preds$fit <- abs(test_log_preds$fit)

correlation_log_accuracy <- cor(test_log_preds)
correlation_log_accuracy
```

```
##           actuals       fit       lwr       upr
## actuals 1.0000000 0.9547039 0.9547019 0.9547059
## fit     0.9547039 1.0000000 1.0000000 1.0000000
## lwr     0.9547019 1.0000000 1.0000000 1.0000000
## upr     0.9547059 1.0000000 1.0000000 1.0000000
```